Report No. 294

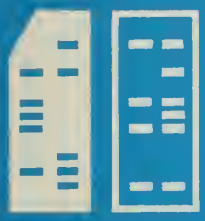
ILLIAC IV

QUARTERLY PROGRESS REPORT

July, August and September, 1968

Contract No.
US AF 30(602)4144

ILLIAC IV Doc. No. 206



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLIAC IV

QUARTERLY PROGRESS REPORT

July, August and September 1968

Contract No.
US AF 30(602)4144

Department of Computer Science
University of Illinois
Urbana, Illinois
61801

November 1, 1968

This work was supported in part by the Department of Computer Science, University of Illinois, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. US AF 30(602)4144.

TABLE OF CONTENTS

1. REPORT SUMMARY

The ILLIAC IV Advisory Committee Meeting was held at Burroughs Corporation, Paoli, Pennsylvania, on July 17 and 18. Personnel from Rome Air Development Center (RADC), Advanced Research Projects Agency (ARPA), Burroughs, Texas Instruments (TI), and the University of Illinois attended the meeting. Also present were the members of the ILLIAC IV Advisory Board and of the Advisory Committee to the National Academy of Sciences on the NIKE-X System.

Technical presentations were made on the system design, the programming activity, the diagnostic system, and the application of ILLIAC IV with phased array radar. Laboratory demonstrations of the breadboard PE and of the thin film memory system were made. The concensus of opinion was that the meetings were excellent and presented the attendees with a detailed report of the ILLIAC IV Project.

The B5500 installation continues to operate well. The use of a dedicated machine has had tremendous impact on the University's effort to develop software. The software effort would have been hindered if the machine were not available on a dedicated basis. Additional magnetic tape drives and disk modules have been installed. Moreover, the predicted load from the design automation and diagnostic effort has resulted in the University ordering an additional processor and I/O equipment.

The data link between Burroughs and the University has been installed, and the Mohawk data terminals provide tape to tape transmission. Design Automation data which is to be run on the B5500 has been transmitted to the University by the data link.

The major diagnostic effort is the checkout of the breadboard PE. Test programs to run on the PE exerciser have been provided by the University. The checkout efforts have been going slower than anticipated, and efforts have been initiated to speed up the checkout of the functional logic.

In software, several areas showed progress. In the ISL translator, improvement in its speed and several program sophistications were made. Changes were made in the syntax preprocessor to increase its efficiency, and the complete documentation of the syntax preprocessor was begun. The CAT language project was divided into four areas to increase its progress and development. The supplementation of Glypnir Version I was, with the exception of procedures, completed. A Version I user's manual is in preparation. Progress was also shown in the syntax and semantic description of SQUASH, the debugging aid to ALGOL.

The ILLIAC IV education consisted of organizing and presenting a course of instruction for ILLIAC IV personnel. The subjects discussed by this course were: ALGOL, the B5500, the ILLIAC IV assembler, Tranquil, and ILLIAC IV applications.

Mr. Richard Stokes of Burroughs has been replaced as Deputy Project Manager by Mr. Walter Fresch. The University has approved the replacement.

The major problems are: Debugging of the PE breadboard and obtaining artwork for the PE and CU printed circuit boards. As mentioned previously, the University will cooperate with Burroughs in the PE debugging by use of the PE logic simulator. The artwork production is being followed very closely, and weekly reports are being received from Burroughs. The University's B5500 facility is available for the Design Automation Activity from 12:00 midnight to 8:00 A.M. nightly, and additional time is provided on weekends. The availability of the computer time and quick turn-around will help the DA effort.

The University has requested Burroughs to use the University's computer to reduce the cost of running the DA programs. The funds from ARPA will carry the project to mid-March.

2. HARDWARE

2.1 Diagnostics

2.1.1 PE Logic Simulator

2.1.1.1 Generation of PE Logic Simulator

The PE Logic Simulator (or the Test Simulator Program) was improved and completed this quarter. The simulator takes two-ten seconds of the B5500 processor time for one clock of the PE.

The simulator body is, basically, a set of ALGOL procedure statements whose identifiers correspond to package types while the actual parameters correspond to the signals incident to the package. There is one procedure statement for each package, and the order of the statements is defined by the level assignment of the packages.

Because there are several looped packages, it is impossible to follow the logic in one sequence; therefore, there is a program loop for each set of the looped packages for evaluating the logic equations until all outputs of the packages are stabilized. If, after thirty times of repetition for the looped packages, the outputs of the packages are not stabilized, a message will be printed out to indicate a possible race condition.

A few problems were caused by the usage of a compiler language as a simulator media. Since the number of words in a program segment cannot exceed 1023, some redundant words as BEGIN, FORMAT, and END were inserted to divide the segment. About four minutes of the B5500 processor time was used to generate the simulator body, which is in a file containing approximately 4000 card images.

An editing program combines the generated simulator body with the procedure declarations and the input/output processing program. The procedure declarations describe the logic of all the package types in ALGOL. The formal parameters of the procedure correspond to sixteen or eighty pins of the package.

The input to the simulator may contain specifications of a microsequence and/or data to be put into the PE and special symbols to control the content of the output from the simulator. The content of every register and the signal values of some combinational circuits, as well as an array showing the state of all the inter-package signals for the convenience of logic debugging, can be printed out by the use of control symbols.

2.1.1.2 Level Assignment and Loop Detection

Before the generation of the simulator body, a level had to be assigned to each package. This was done by referring to the arc list which is a reduced form of the wire list. Since the level is not assigned to a package by the logical equations but by the arc list, some packages constitute a loop; therefore, to assign a meaningful label to all packages, the steps listed below should be followed.

I. Assign the level to the packages in the ascending order and in the descending order. Extract the packages whose levels are not defined -- these packages may constitute loops.

II. Distinguish each loop from the set of extracted packages. Looped packages are reduced to one pseudo-package.

III. Assign the level to the set of pseudo-packages, and after assigning the level, expand each pseudo-package to the original packages.

All packages are converted to the compact symbolic names within these programs to increase the speed of level assignment. There are several programs for this conversion.

The simulator can treat up to 2000 arcs and 500 packages with up to 1000 arcs involved in loops. There are about 1800 arcs and about 450 packages (including some dummy packages) in the PE. The number of arcs within the looped packages is 156 and the number of packages is 482. The processor time for these programs is four minutes for the first step, thirty minutes for loop detection, and four minutes for the last step.

2.1.1.3 Application to Logic Debugging

The simulator has been used to debug the logic design and the wiring list of the PE. The basic transmit and arithmetic instructions were tested on the simulator, and Burroughs was informed of the detected errors.

2.1.2 Generation of PE Diagnostic Programs

2.1.2.1 Path Tests

The last two subprograms of the Test Ordering Program (TØP) were written and debugged. The algorithm used in the ordering of test cases is based on the evaluation of a weight for each test. It has a tendency to prolong the chains of "success" branches compared with those of "failure" branches in the test tree. This algorithm is expected to be more powerful than other methods in locating multiple errors.

The following are major files generated by the TØP.

I. TØP/NBRTWIG: It contains an integer which indicates the size of the following three files. At present this integer is 831.

II. TØP/PTHNAME: This file specifies the path to be tested or the failure location (at the termination). This file is a one-dimensional array.

III. TØP/SUCBRNH: It is a one-dimensional array. The value tells where to branch if it is non-zero. Zero indicates a termination of the tests.

IV. TØP/NODELBL: This file is also a one-dimensional array, and it indicates the branched-to location. Some tests are labeled.

The path test sequence can be specified by the previous four files. Some additional information about failure locations (e.g., equivalent failure location and set of equivalent multiple failure locations) should be referred to when the path test sequence is used. An additional program is being written to combine these

files into one file having the format of the PE Exerciser Test Assembly language.

The output of the Path Test Generator has been reviewed In the input and in the program, a few errors were removed.

2.1.2.2 Combinational Tests

Some final additions and modifications were made to the Combinational Test Generator during this quarter. The expected response procedure for BSW was added, and the procedure for CSA was modified slightly to perform the expected response calculations for MSG and MDG tests. All expected response procedures are now operating, and few additional modifications are anticipated. The documentation for CTG was also completed during this quarter.

Several programs were written which translate the output of CTG (in PEX assembly language) into a form suitable for input to the PE simulator. An old version of the PEX Input Generator (PIG) was rewritten to facilitate the use with the combinational test record. This program translates control signal names into a fixed format for the PEX Test Assembly Program (PEXTAP). Another program, Assembled Code Translator, written during this quarter is used to convert the machine code output of PEXTAP into the form required by the simulator.

With these three programs (PIG, PEXTAP, and ACT), any program written in the PEX assembly language can be run on the PE simulator. Running of sample combinational test programs provides a means for checking the programs themselves, for checking the expected response calculations in CTG, and for possibly detecting design errors in hardware.

A large sample of the CPA and ADA tests was run on the simulator. These runs indicated that the CPA tests, CPA expected response calculations in CTG, the simulator, and the PE were all functioning properly. In the case of ADA, the expected response calculation in CTG and the polarity of one signal in the PE were found to be in error. With these errors corrected, the address

adder tests, CTG, simulator, and PE were also in agreement for ADA. Presently, a sample of the Barrel Switch tests is being run. It is hoped that future runs of these programs will aid Burroughs in its debugging efforts and will serve to eliminate any errors in the combinational test programs and the expected response calculations.

2.2 Design Automation

Final specifications for the Delay Check Program were completed in the first part of this quarter. The program was written and tested on small test boards here at the University. Final debugging and testing are held up until the Post-Processor Program is operational. The work to make this program operational will continue into the next quarter.

Arrangements which will enable Burroughs DAS production runs to be done on the B5500 here at the University are nearly complete. A small software package has been written to convert program data files into tapes for the Mohawk Data Set and for their reconversion. This operation should increase the use of the B5500 here at the University and give the Burroughs group added machine capability.

Work has begun on a preprocessor program for an ILLIAC IV Design Automation System. This program will input the data to the program, convert it into a suitable data file, and check for user syntax errors. The program will also prepare useful reports for aiding a designer in verifying his program input, logic diagrams, and equations.

3.1 Translator Writing System and Language Development

3.1.1 Introduction

Progress was shown in many areas during this quarter. Some of the progress involved changes which increased the efficiency in programs and the development of a new language. Improvements were made in the parser, and there was also progress in converting the parser instruction table to an ALGOL program. The ISL translator, during this quarter, was improved; and the Pass II of the Tranquil compiler was begun. The following paragraphs discuss these and other areas of progress.

3.1.2 Syntax Preprocessor

The implementation of minor changes in the syntax preprocessor continued. The purpose of these changes is to increase the preprocessor's efficiency. A line by line rewriting of the code which is also taking place will increase efficiency.

The inception of complete documentation of the syntax preprocessor has begun. The initial emphasis of this documentation is the attainment of a detailed description of the algorithm used in developing the syntax preprocessor.

3.1.3 Parser

A much more efficient version of the parser was implemented and debugged. All the indirect addressing was replaced by direct transfer addresses. The parser instructions were expanded to six bits allowing new, more efficient parser instructions to be generated and allowing the use of stream procedures for fetching or testing parser instructions and operands. The new parser has run from 100 to 750 cards per minute on test programs of various complexity.

The new parser which implements more meaningful error and monitoring messages has an expanded error recovery scheme which actually corrects errors in certain situations. By having many tests and sorts performed in the syntax preprocessor, the testing done at parse time is kept to a minimum.

Work is nearly finished on a program that will convert the parser instruction table to an ALGOL program. This should add another significant increase to the speed of the parser since most procedure calls and array accesses will be eliminated by this approach.

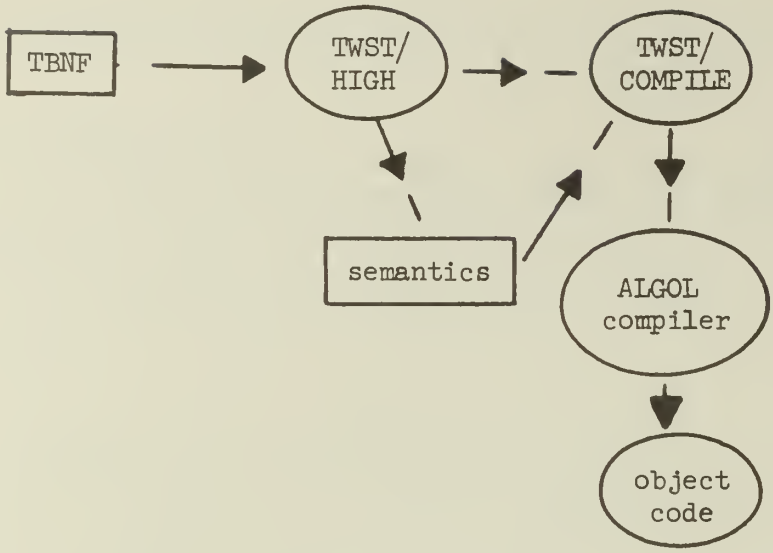
3.1.4 Twinkle

Another area of effort was the description and implementation of Twinkle, a syntax description language. It will not only combine all the features of present syntax languages used by the ILLIAC IV Translator Writing Systems but will also contain additional features. A TWS generated recognizer for this language will become the syntax scanner for the syntax preprocessor.

3.1.5 TWST/TBNF

A new version of TWST which translates from TBNF (translatable BNF, i.e., extended BNF) to Burrough's code has been completed and seems to be reliable. It offers a two-three fold speed increase over the table driven version. A precedence notation compatible with the scheme has been planned but not yet implemented. Tapes and programming manuals for TWST/TBNF are available.

Schematic diagram



3.1.6 TWS Semantics - ISL Translator

During this quarter, a high priority was given to the improvements in the brute-force ISL translator in order to make the task of the high-level language groups easier. The speed improvement effort was successfully completed. Several program sophistications were introduced in the translator which resulted in a speed improvement by a factor of six. The present brute-force translator can now operate, on the average, at 1200 cards per minute of processor time. Also completed was the addition of control cards to the translator.

Work is now continuing in two areas: 1.) The TWS-ISL translator is being completed, and pass-n facilities are being added to both translators; 2.) Being started is a documentation effort whose goal is to produce, as soon as possible, an ISL user's manual containing examples of semantic descriptions of programming languages.

3.2 Tranquil

The Tranquil work for this quarter involved three areas. One of the work areas concerned storage allocation in Tranquil. In the instances that all data fit in PE memory, the compiler accepts them and allocates the spaces for them. The array type data is cut into blocks of size 256 x 256. Operations and I/O between the disk and the memory are done in terms of these blocks.

The compile time allocation of space in the CU was another effort area. Routines were written for the compile time allocation of space in the CU and further work was done on the use of storage schemes for sets. The CU allocation routines involve a priority system and a compile run-time stack and include CAR and LDB usage. The set schemes are linked with their declarations, and the amount of information given or found determines whether dynamic allocation needs to be incorporated.

The third area of Tranquil effort involved the beginning of the major programming effort for Pass II. The overall structure of Pass II was laid out and programmed. Partially completed were programs to analyze assignment statements for their meaning and for compiling. The algorithm for determining non-dynamic variables was implemented.

3.3 Glypnir

Implementation of Glypnir Version I was, with the exception of procedures, completed during the third quarter of 1968. Procedures will be implemented during the fourth quarter. The code generated by the compiler has been debugged syntactically on the current version of the assembler, and complete debugging will begin as soon as the new ILLIAC IV simulator becomes available. A Version I user's manual is in preparation, and copies of the rough draft can be obtained.

3.4 SQUASH

The syntax and semantic description of SQUASH, the debugging aid to ALGOL, has progressed. The syntax is virtually complete and

has been successfully accepted by SYNPROF, the TWS syntax processor. However, minor modifications in syntax may be necessary to facilitate the writing of semantics. But, having considered semantics, coding has begun.

3.5 System K

3.5.1 Introduction

The main activities of the SYSK group this quarter concerned many areas. These areas were the ILLIAC IV assembler, simulator and loader, design of the B6500 operating system, and development of OSK prototype features.

3.5.2 Assembler

A new version of the assembler was written to serve as the foundation of the forthcoming macro assembler. Much time was spent in making this assembler as fast as possible. The assembler is substantially complete. It emits pseudo-orders for the loader; it accepts the latest ILLIAC IV order code; it creates a cross-reference table; and it has extensive file manipulation facilities to aid the programmer in maintaining large programs. Extensions are now being added for TMU (the Test Maintenance Unit on the ILLIAC IV) commands. This assembler with the TMU command mode will be sufficient for the B5500 ILLIAC IV operating system at Paoli, Pennsylvania.

3.5.3 Simulator

The development of an efficient, single quadrant ILLIAC IV simulator was slowed because all the operation code assignments and the instruction semantics were changed. But the simulator coding is now substantially complete. A timing simulator was written and will be integrated into the simulation package. Simulator maintenance will be handled by a new individual so that the presently involved person is free to work full time on the loader. The transition will occur on the first of October.

3.5.4 Loader

The basic functions of the ILLIAC IV loader have been defined. In particular, the loader related code information (address relocatability, storage assignment, external references, etc.) is now fixed, and its fields and values in the object code file assigned. The new assembler emits this loader information, and the new simulator includes a loader to interpret this information.

3.5.5 OSK (ILLIAC IV Operating System) Development

There was considerable discussion of the alternative means of handling IOC interrupts on the B6500 and of the structure of the B6500 control programs. The write-up of the summarization of the decisions is in preparation. During the next quarter, a prototype version will be coded to work with the new simulator.

3.5.6 Interim OSK Features on the B5500

Progress is being made on integrating ILLIAC IV languages with the B5500 system. The deck needed for a simple assemble and execute run is listed below.

```
?USER = LP
?COMPILE  LP/TEST WITH ASK LIBRARY
?DATA CARD
 ____
 ____      ASK assembly program
 ____
 ____
?EXECUTE  LP/TEST
?END
```

Work is continuing on a simulator scheduler for the B5500. The goal of this project is to call the simulator into execution whenever the B5500 is not busy on foreground tasks. No operator will be required because the simulator initiation will be automatic. Also, no user will be able to destroy the queue of simulations since the

scheduler will survive system hang-up, halt loads, and any disaster that does not destroy the disk (so far no user has ever destroyed the disk). Simulations will be scheduled using a prototype user services subsystem. The scheduler will work with any other long-running job that is suitably halt-load proofed.

3.6 CAT

3.6.1 General Compendium

The CAT language project has been divided into four distinct efforts whose results will fit into the ILLIAC IV system in various places. This is a result of meetings that were held here this summer with groups of users. The four effort areas are as follows: 1) general I/O routines which will add to Tranquil some kind of disk read and write statements using symbolic file names; 2) a descriptive geometry language which will allow the user to specify his problem space in a general notation rather than forcing him to tediously define every index set; 3) a study of storage allocation schemes for both fast memory and disk with a view toward implementing general techniques within a compiler (a possible result may be automatic I/O which makes the ILLIAC IV disk appear to be an extension of memory); and 4) a study of the optimization of disk I/O by linear programming techniques which reorder the arithmetic statements within a code to allow efficient use of the data set.

In addition to the usefulness of each of these studies as an individual part of the ILLIAC IV system, it is hoped that they can be integrated into a software package for the general user. In this line, a study is also being made of the non-mathematical parts of several large programs; since such "data processing" seems to account for as much as eighty percent of the total run time in many computing centers, an efficient and general system on ILLIAC IV is evidently highly useful.

3.6.2 General Optimization

In the fourth area mentioned above, work began this quarter. This work is involving an attempt to minimize disk latency for non-core contained ILLIAC IV problems. In so doing, permutation of source code, as well as dynamic relocation of data on the disk, is being considered.

At present, the method of approach involves quantizing the time axis over which any computation will take place. Binary variables are then introduced which have values corresponding to each of the actions taken (e.g., compute, input, overlay, etc.) during the particular time interval. It is hoped that, with only a few reasonable simplifying assumptions, constraints may be placed upon these binary variables. The difficulty is keeping these constraints linear.

4.1 Mathematical Applications

4.1.1 Partial Differential Equations

During this quarter, a research assistant spent six weeks at Los Alamos Scientific Laboratory, Los Alamos, New Mexico. A Tranquil code was written for the Particle in Cell (PIC) method used in hydromagnetic theory. While at Los Alamos, much machine time was used for collecting statistics on various storage schemes for particle in cell methods as applied to a parallel computer. The future plan is to fully document this Tranquil code and the collected statistics during the next quarter.

4.1.2 Ordinary Differential Equations

Work was completed on a system of equations for metabolic systems, and this system was summarized in ILLIAC IV Document 197 [1]. The assembly code for this problem is being kept up to date as newer versions of the assembler are available.

Further study is being done on a system of equations related to the physics of a muonic atom. These equations are:

$$\frac{dF_j^K(r)}{dr} = \frac{K_j F_j^K(r)}{r} + (B + U_0(r) + W_j)\ G_j^K(r)$$

$$+ U_2(r) \sum_i A_{ji}\ G_i^K(r)$$

$$\frac{dG_j^K(r)}{dr} = \frac{-K_j G_j^K(r)}{r} + (2 - B - U_0(r) - W_j) F_j^K(r)$$

$$+ U_2(r) \sum A_{ji} F_i^K(r)$$

The object is to find an exact value for the eigenvalue B. Given an approximation to B and suitable boundary conditions, integration takes place from the inner and outer boundaries to a central "fitting radius". The match of the results at this radius determines the new value of B for the next iteration of this process.

For the large problems in this area, the number of equations is larger than the number of PE's available, and a method must be devised for sharing the computation of the equations over these PE's. Assembly codes are being written to test different but like strategies and to do the actual computation.

4.1.3 Alternating Direction Iteration Scheme

In these three months, an alternating direction iteration scheme in ASK was written, compiled, and time simulated. The time simulation consists of running the program for one and two iteration cycles on the Sankin Time Simulator and separating the $I/O \sim 1.10 \times 10^{-3}$ secs from actual iteration time $\sim 2.3 \times 10^{-3}$ secs.

The code is now in the final stages of simulation, in comparison to the same code in ALGOL, and in documentation. The solution matrix has a small oscillation in it, otherwise the simulation is complete. Two important facts will be the result of the comparison of this code to an identical ALGOL code. One, it will check our results in computation, and two, it will give a comparison of the time simulation to conventional machines. The written report will follow the completion of these two remaining problems.

Also, a successive, over-relaxation iteration scheme in Tranquil has been compiled. This scheme is presently being rechecked for computational errors.

4.1.4 Hydrodynamic Codes

The numerical solution of the Eulerian hydrodynamic equations, in two-dimensional cartesian coordinates, was coded in Tranquil using checkerboard storage. An additional code which traces the path of

"mass-less" particles through an Eulerian grid was also completed during this quarter. Both codes have been syntactically debugged for the Tranquil compiler. An ILLIAC IV Document which will describe the methods used and the codes will appear soon.

4.1.5 Boltzmann's Equation

The ILLIAC IV Document 200 [2] describing the implementation of a Monte Carlo method for evaluating the Boltzmann collision integral to ILLIAC IV was completed during this quarter. The method that is being used for the evaluation of the Boltzmann collision integral was developed by Arnold Nordsieck and Bruce L. Hicks of the University of Illinois [3]. The implementation of this method for ILLIAC IV requires the generation of random numbers in each PE which are also random with respect to the random numbers being generated by the other PE's. Different random number generators for ILLIAC IV which will insure this requirement are being considered. A Tranquil code will be written for this method of evaluating the Boltzmann equation.

4.1.6 Matrices

An algorithm to find the solution matrix X to a symmetic matrix A by using the square-root method (Cholesky [4]) was coded in Tranquil. The order of the matrix A was $n = 64$, but it can without any difficulty be extended to any magnitude $n \leq 256$.

From theory it is known that a symmetric matrix can be written in the form $A = SS^T$. Using the upper triangle of A, the transpose S^T is then given by

$$s^T_{11} = \sqrt{a_{11}} \quad , \quad s^T_{ij} = \frac{a_{ij}}{s_{11}} \quad j > 1.$$

Further, $s^T_{ii} = \sqrt{(a_{ii} - \sum_{\ell=1}^{i-1} s^2_{\ell i})}$ $\quad i > 1$

$$s^T_{ij} = \frac{a_{ij} - \sum_{\ell=1}^{i=1} s_{\ell i}\, s_{\ell j}}{s_{ii}} \qquad j > 1 \text{ and } i < j$$

$$s^T_{ij} = 0 \qquad i > j.$$

With $A = SS^T$ and $Ax = b$, the result of substitution is

$$SS^T x = b.$$

From this $S^T x = y$ and $Sy = b$.
To find y from the lower triangle S, the following formulas are used.

$$y_1 = \frac{b_1}{s_{11}} \quad , \quad y_n = (b_n - \sum_{\ell=1}^{n-1} s_{n\ell}\, y_\ell) \mid s_{nn} \quad , \quad n > 1 .$$

Having y, x is found by using the upper triangular matrix S^T and back substituting:

$$x_n = \frac{y_n}{s^T_{nn}} \quad , \quad x_k = (y_k - \sum_{\ell=k+1}^{n} s_{k\ell}\, x_\ell) \mid s_{kk} \quad , \quad k < n.$$

The problem of having a negative argument in determining the roots in S^T has been taken care of by using the marker "-" (minus) with the rule in mind that "-" × "-"="-".

Then s^T_{ii} becomes:

$$s^T_{ii} = - \sqrt{|a_{ii} - \Sigma\ s^2_{\ell i}\ |} \quad .$$

Considering the formulas above, a "straight" storage scheme for all matrices involved is suggested. To illustrate this

point, let us look at an 8 x 8 matrix S^T. Assuming the elements $s_{ik} \in S^T$ with k=i, i+1, ..., 8 for $i \leq 3$ have been calculated, then the following is the case:

$$\begin{matrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} & s_{16} & s_{17} & s_{18} \\ & s_{22} & s_{23} & s_{24} & s_{25} & s_{26} & s_{27} & s_{28} \\ & & s_{33} & s_{34} & s_{35} & s_{36} & s_{37} & s_{38} \\ & & & s_{44} & (s_{45}) & & & \end{matrix}$$

With s_{44} already known, the calculation of s_{45} is wanted.

$$s_{45} = [a_{45} - (s_{14}\, s_{15} + s_{24}\, s_{25} + s_{34}\, s_{35})] \mid s_{44}$$

The elements above s_{44} and s_{45} are involved in this calculation, or stated more generally, it is: To calculate s_{ik}, $k > i$, only the elements of columns i and k which are above s_{ii} and s_{ik} enter the calculation. Furthermore, since the program has been written in such a way that the elements in row i, which are to be determined, are under SIM control, the efficiency of a "straight" storage scheme becomes even more apparent.

4.1.7 Eigenvalues

During this quarter, work was done in the investigation of eigenvalue problems. A code in assembly language has been written for Jacobi's Method for finding eigenvalues. The algorithm used in the code is a modification of the classical Jacobi Method. The matrix is divided into 2 x 2 sub-matrices along the diagonal, and successive orthogonal transformations are used to eliminate the off-diagonal elements of each sub-matrix. With each iteration, n elements of an n x n matrix are eliminated. The code is presently being debugged, and various storage schemes are being considered. The code is being

timed both on the SANKIN Simulating System and on the SIM/TIME Simulator. A timing estimate will also be derived from the IBM 360 for purposes of comparison.

4.1.8 Root Finding

During this quarter, work continued on adapting Lehmer's algorithm to a parallel machine. Lehmer's algorithm determines whether or not a polynomial has a root in a given circle. Using Gerschgorin circles, the center and the radius of a circle in which all of the roots of a polynomial are found can be obtained. The problem is how to efficiently cover this area with 256 circles so that the percent of duplication is minimal.

The first approach was to divide the area obtained by Gerschgorin circles into 256 identical squares. This area was then covered by circumscribing circles around the squares. The next time, this area was covered by inscribing circles, and the missed area was covered with other circles. In both of these cases, the overlap was fifty-seven percent.

To reduce this overlap figure, the second approach was to divide the area into 256 identical hexagons. In this case, only circumscribing circles were tried, and the duplication was twenty-one percent. Future work will be directed toward reducing the overlap figure more.

4.1.9 Special Functions Subroutine Library

This quarter, work has continued in ILLIAC IV's special functions subroutine library. All previously coded subroutines have been recoded in the latest version of the ASK assembly language. The question of subroutine linkage was also considered in the rewriting of these codes. It was decided that the function argument should be in the A register upon entering the subroutine, and its evaluation would be left in the A register. It was also decided that the subroutine itself would save all of the other registers, except the B register, and restore these at the end of the routine. The functions

will be evaluated only in the enabled PE's, and all other PE's will be left alone.

New 64-bit codes have been written for natural logarithm and arctangent. Also, a new 64-bit code has been developed for square root which completely eliminates division.

I. Natural logarithm: Let y be the number of which it is desired to find the natural logarithm.

Then

$$y = 2^i m \qquad \frac{1}{2} \leq m < 1$$

where i is the floating point number exponent and m is the floating point number mantissa.

The approximation of $\log_2 m$ is made by a polynomial for $\frac{1}{2} \leq m < 1$. Then the natural logarithm is evaluated in accordance with the relation:

$$\ln y = (i + \log_2 m) \cdot \ln 2$$

II. Arctangent: The approximation of arctan (x) is made by a polynomial for $x \in [0, \text{Tan } \pi/8]$.

For $x \in (\text{Tan } \pi/8 , \text{Tan } 3\pi/8]$

$$\text{Arctan } (x) = \text{Arctan } (^{x-1}/_{x+1}) + {}^{\pi}/_{4}$$

For $x \in (\text{Tan } {}^{3\pi}/_{8}, \infty)$

$$\text{Arctan } (x) = \pi/2 - \text{Arctan } ({}^{1}/_{x})$$

The above three cases all use the same approximating polynomial.

III. Square Root: Let A be the number of which it is desired to find the square root. The iterative scheme used is:

$$x_{n+1} = x_n/2 \ (3 - A\, x_n^{\,2})$$

to approximate $1/\sqrt{A}$

Thus $\sqrt{A} = A \lim_{n\to\infty} x_n$

Starting points for this iteration are approximated by second degree polynomials.

4.1.10 Long Codes

As a first step in this task, preliminary studies were done of the Theory of Stability of Motion and of Canonical Transformations. The behavior of the solutions for equations of motion having the form $\dot{x} = Ax$ was investigated. (It was assumed that A was a constant coefficient matrix.) Also, the criteria of stability of the above-mentioned autonomous systems were reviewed. Finally, a detailed illustrative example was worked out for showing the effect of errors in the observation of the initial vector x_0 on the solution of the equations of motion for a given autonomous mechanical system.

4.2 Linear Programming

During this quarter, specification of the mathematical procedures for the first linear programming system, LPS, has largely been completed. Procedures for handling vector bounds, ranges of the right hand side, and basic parametric programming have been drafted. Also during the quarter, the group has examined applications of linear programming to problems of hardware design, along lines suggested by Dr. Masao Kato during his visit to the University.

The solution of a linear programming problem in standard form is handled by a modification of the revised simplex method, product form. The original matrix is partitioned to produce a degree of parallelism suitable for ILLIAC IV. Rows are assigned to specific PE's in a manner designed to distribute calculations evenly among the PE's. The matrix is skewed within and across quadrants to facilitate vector updating and reduced cost calculations. Multiple pricing has been adopted to minimize the number of iterations and disk accesses, thereby reducing overall calculation time and increasing accuracy. Tests have been initiated to evaluate the efficiency of the algorithm with regard to PE utilization and storage allocation.

It is necessary at certain points in the solution procedure to recalculate the updated inverse in product form, minimizing the number of non-zero elements obtained while maintaining a high degree of accuracy. Work continues in the development of such a reinversion procedure from the several techniques which have been examined.

4.3 Radar Processing Applications

Some of the efforts during these three months have been involved in the conversion of the Kalman Filter tracking programs to 32-bit floating point mode. The development of a Tranquil version of the Kalman Filter and the analysis of the NISIM (NIKE Simulation) programs from Bell Telephone Labs were other areas involved in this quarter's efforts.

The assembly language version of the Kalman Filter tracking algorithm was completely recoded so that it would run in 32-bit floating point mode instead of 64-bit fixed point. These programs will handle the correlation of newly received data to locate the track table, start a new table for new targets, and move tables if required because of target changing drastically in azimuth and elevation. Also, they will perform the Bayesian estimation and will integrate the dynamic equations of motion over time for prediction and the coordinate conversions. These programs total, approximately, 4500 instructions and have been run on the timing simulator; however, they have not been run on the ILLIAC IV execution simulator because, at present, the simulator is not complete.

The main sections of the Kalman Filter programs-namely the Bayesian estimation and the integration of the dynamic equation of motion-have been programmed in ALGOL for the B5500 computer. These B5500 programs will be used to generate simulated tracking numbers for debugging the ILLIAC IV Kalman programs. The debugging of these programs will begin as soon as the execution simulator is operating. Also, the Tranquil version of the Kalman Filter tracking program will be debugged and run when the compiler and execution simulator are finished. By comparing the assembly language version and the Tranquil version of the program, it will give an evaluation of how efficiently these BMD problems can be programmed in the higher level language (Tranquil) for ILLIAC IV. This can test both the flexibility of debugging Tranquil programs over assembly language programs and the operating time penalty which is paid for using the higher level language. A report will be generated in the near future which will fully describe the Kalman Filter for ILLIAC IV.

Over this time period, there has been a slowdown in progress due to changes in personnel on this effort, and this slowdown will continue for the next time period. The two graduate students working on this effort have left the University. Presently, the effort has one full time professional, one part-time hourly, and one or two new graduate students who are just starting and require time to become familiar with ILLIAC IV.

A better understanding of the NISIM programs is being obtained in order to evaluate how this type of BMD programs would fit on ILLIAC IV. More analysis and a better understanding of them is still required before a complete sample BMD type problem can be set up for the ILLIAC IV computer.

4.4 ILLIAC IV Education

The ILLIAC IV Education for July and August consisted of organizing and presenting a course of instruction for ILLIAC IV personnel. Many newcomers were initially confused and swamped with the enormity of the ILLIAC IV project, but toward the end of the semester most of them were well oriented and producing useful work. The course was basically divided into three sections, and a brief discussion of each follows.

I. ALGOL and the B5500 -- The basic concepts of ALGOL and the special B5500 ALGOL features were presented. A detailed description of how to get programs running on the various B5500 hardware devices proved extremely valuable.

II. ILLIAC IV Assembler -- The Assembler was explained and examples given of the various types of orders, and case studies were examined. Several competent assembly language programmers emerged by the end of the summer.

III. Tranquil and ILLIAC IV Applications -- The Tranquil lectures had to be mostly theoretical because no facility for testing Tranquil code existed at the time of the course. General discussion of applications gave an insight into the scope and use of ILLIAC IV.

Work continued on compiling a useful programming manual for the Assembler. Parts of this have been completed and distributed to those requiring the information.

REFERENCES

[1] McCarthy, Thomas. Solution of Ordinary Differential Equations Related to Metabolic Systems. ILLIAC IV Document Number 197, (July 11, 1968).

[2] Winje, G. L. Implementation of the "Monte Carlo Evaluation of the Boltzmann Collision Integral" on ILLIAC IV. ILLIAC IV Document Number 200, (August 1, 1968).

[3] Nordsieck, Arnold, and Hicks, Bruce L. Monte Carlo Evaluation of the Boltzmann Collision Integral. Coordinated Science Laboratory Report R-307, (July, 1966).

[4] Faddeeva, V. N. Computational Methods Of Linear Algebra. New York: Dover Publications, Inc., 1959. p. 81 ff.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author) Department of Computer Science University of Illinois Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

REPORT TITLE

ILLIAC IV QUARTERLY PROGRESS REPORT
July, August and September 1968

DESCRIPTIVE NOTES (Type of report and inclusive dates)

Progress Report

AUTHOR(S) (First name, middle initial, last name)

REPORT DATE November 1, 1968	7a. TOTAL NO. OF PAGES 30	7b. NO. OF REFS 4
a. CONTRACT OR GRANT NO. 46-26-15-305 b. PROJECT NO. USAF 30(602)4144 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

0. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DCS.

1. SUPPLEMENTARY NOTES NONE	12. SPONSORING MILITARY ACTIVITY Rome Air Development Center Griffiss Air Force Base Rome, New York 13440

3. ABSTRACT

See the report summary within the Report itself.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Twinkle TWST/TBNF						